A2-Q3: Parametric Spline

In [1]:
```
import numpy as np
from scipy.interpolate import make_interp_spline
import matplotlib.pyplot as plt

%matplotlib qt
plt.ion()
```

(a) Write your nickname and display it

In [2]:
```
# [1] Display nickname image
plt.figure(1)
f = plt.imread('nickname.jpg'); plt.imshow(f);
c = plt.ginput(100, mouse_stop=2, timeout=120)
plt.draw()
```

(b) Hardcode interpolation points

In [7]:
```
c
```

Out[7]:
```
[(254.9838709677419, 208.76451612903236),
 (261.95161290322574, 278.441935483871),
 (252.6612903225806, 378.3129032258065),
 (306.08064516129025, 238.95806451612918),
 (336.274193548387, 297.0225806451614),
 (299.1129032258064, 355.0870967741936),
 (501.1774193548386, 215.73225806451626),
 (489.56451612903214, 287.73225806451626),
 (466.3387096774192, 359.73225806451626),
 (524.4032258064515, 371.34516129032266),
 (573.1774193548385, 371.34516129032266)]
```

In [28]:
```
x1 = [254.9838709677419, 261.95161290322574, 252.6612903225806]
y1 = [208.76451612903236, 278.441935483871, 378.3129032258065]

x2 = [254.9838709677419, 306.08064516129025, 336.274193548387, 299.1129032258064, 252.6
y2 = [208.76451612903236, 238.95806451612918, 297.0225806451614, 355.0870967741936, 378

x3 = [466.3387096774192, 489.56451612903214, 501.1774193548386]
y3 = [359.73225806451626, 287.73225806451626, 215.73225806451626]

x4 = [466.3387096774192, 524.4032258064515, 573.1774193548385]
y4 = [359.73225806451626, 371.34516129032266, 371.34516129032266]
```

(c) ParametricSpline

In [35]:
```
def ParametricSpline(Sx,Sy):
    '''
     x_cs, y_cs, t = ParametricSpline(Sx,Sy)

       Takes an array of x- and y-values, and returns a parametric
       cubic spline in the form of two piecewise-cubic data structures
       (one for the x-component and one for the y-component), as well as
```

```
    the corresponding parameter values.

    The splines use natural boundary conditions.

    Input:
     Sx   array of x-values
     Sy   array of y-values

    Output:
     x_cs function that evaluates the cubic spline for x-component
     y_cs function that evaluates the cubic spline for y-component
     t is the array of parameter values use for the splines

    Note that x_cs(t) and y_cs(t) give Sx and Sy, respectively.
    '''
    N = len(Sx)
    t = np.arange(N)
    x_cs = make_interp_spline(t, Sx, bc_type=([(2, 0.0)], [(2, 0.0)]))
    y_cs = make_interp_spline(t, Sy, bc_type=([(2, 0.0)], [(2, 0.0)]))

    return x_cs, y_cs, t
```

(d) Find parametric splines for each segment

In [36]:

```
i1 = ParametricSpline(x1, y1)
t1 = np.arange(len(x1))
tt1 = np.linspace(t1[0], t1[-1], 1000)
xx1 = i1[0](tt1)
yy1 = i1[1](tt1)

i2 = ParametricSpline(x2, y2)
t2 = np.arange(len(x2))
tt2 = np.linspace(t2[0], t2[-1], 1000)
xx2 = i2[0](tt2)
yy2 = i2[1](tt2)

i3 = ParametricSpline(x3, y3)
t3 = np.arange(len(x3))
tt3 = np.linspace(t3[0], t3[-1], 1000)
xx3 = i3[0](tt3)
yy3 = i3[1](tt3)

i4 = ParametricSpline(x4, y4)
t4 = np.arange(len(x4))
tt4 = np.linspace(t4[0], t4[-1], 1000)
xx4 = i4[0](tt)
yy4 = i4[1](tt)
```

(e) Plot the segments

In [37]:

```
plt.figure(figsize=(8,8));
plt.gca().invert_yaxis()
plt.plot(xx1, yy1)
plt.plot(xx2, yy2)
plt.plot(xx3, yy3)
plt.plot(xx4, yy4)
plt.plot(x1, y1, 'ro');
plt.plot(x2, y2, 'ro');
```

```
plt.plot(x3, y3, 'ro');
plt.plot(x4, y4, 'ro');
plt.axis('equal');
plt.xlabel('x'); plt.ylabel('y');
```

In [ ]: